1 Introduction

In this document, we will attempt to create a face image classifier by using various machine learning models. All models are created from scratch using MATLAB. The models being discussed are a Gaussian model, a Gaussian mixture model (GMM), a Student's t-Distribution model, a Student's t-Distribution mixture model, as well as a factor analysis model. Images being read are in the JPG format, vary in size but are always square, 3 RGB channels, using the face database founds here - (http://vis-www.cs.umass.edu/fddb/).

The goal is for all these models to take an image $\mathbf{x}$ and assign a label z to it, where $z \in \{0, 1\}$ to determine whether the sampled image is a face or not. We apply Bayes' rule to result in this simple expression for classifying each image

$$Pr(z = 1 \mid \mathbf{x}) = \frac{Pr(\mathbf{x} \mid w = 1)Pr(w = 1)}{\sum_{i=0}^{1} Pr(\mathbf{x} \mid w = i)Pr(w = i)} \tag{1}$$

We will use an arbitrary threshold, h to round $Pr(z = 1 \mid \mathbf{x})$ up or down to conclusively determine whether it is a face or not ($Pr(z = 1 \mid \mathbf{x}) > h \rightarrow$ face). Each model will also have an receiver operating characteristic (ROC) curve, where we sweep h to determine the number of true and false positive rate.

2 Gaussian Distribution as marginalization

2.1 Algorithm

A single Gaussian distribution can be used to model our image data by estimating the likelihood function which maximizes our parameters (MLE). The Gaussian, or Normal, distributions probability density function (pdf) is defined as

$$p(x) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x_n - \mu)^2}{2\sigma^2} \tag{2}$$

where x represents vector sized N of sampled data x_n, μ represents the mean of the sampled data, σ^2 represents the variance of the dataset, and $x \in \mathbb{R}^n$, $\{\mu, \sigma^2\} \in \mathbb{R}$. The MLE of the Gaussian pdf is found by maximizing the likelihood function w.r.t. the input parameters μ and σ^2. We take the pdf, set the partial derivatives w.r.t. each parameter to zero, then solve. The MLE of the mean results in

$$\hat{\mu} = \frac{\sum_{n=1}^{N} x_n}{N} \tag{3}$$

whereas the MLE of the variance results in

$$\hat{\sigma}^2 = \sum_{n=1}^{N} \frac{(x_n - \hat{\mu})^2}{N} \tag{4}$$

We can apply these results to each individual pixel within the images to prevent the need to increase the dimensions of our model. Therefore, the single Gaussian model tells us the sample mean of the images gives the best estimator for the true mean. However, a our best estimator for the variance does not equal the sample variance $s^2 = \sum_{n=1}^{N} \frac{(x_n - \hat{\mu})^2}{N-1}$.

2.2 Results

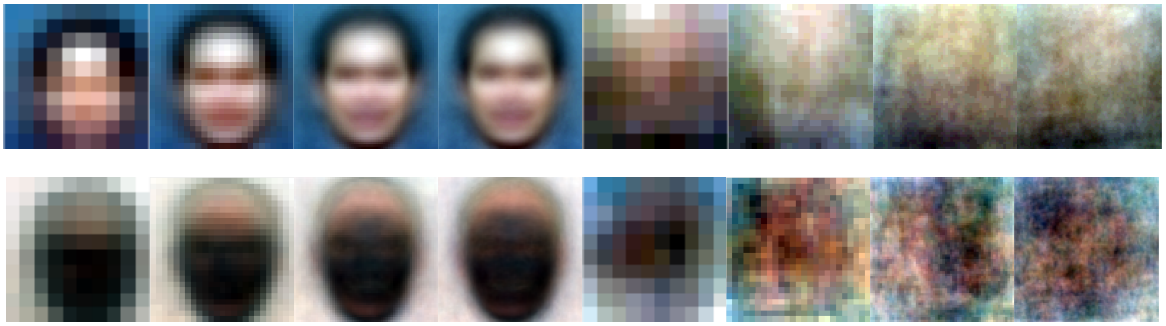

Figure 1: Means and diagonal covariances for $n \times n \times 3$, $n \in \{10, 25, 60, 100\}$, trained on face and non-face images for a single Gaussian model

The reason why we can go up to $n = 100$ is due to the fact that the parameter for $\mathbf{\Sigma}$ which maximizes the likelihood function happens to be the diagonal of the covariance matrix. Meaning, that instead of attempting to calculate the entire covariance matrix, we can simply do operations element-wise for the covariance calculation.

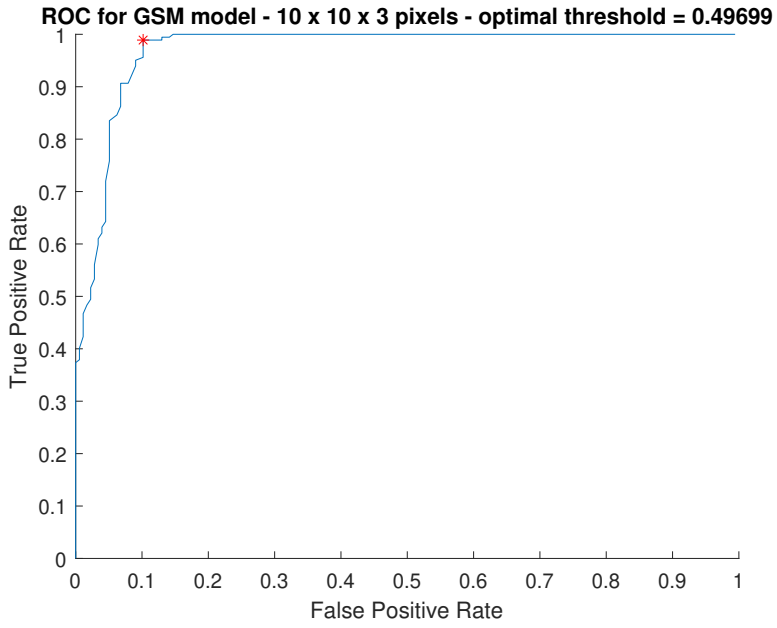

Figure 2: ROC for the $n = 10$ image Gaussian single model

The Gaussian single model with resolution 10 had a surprisingly good ROC curve. If many images are pixelated to the this point, this model seems to be fairly promising.

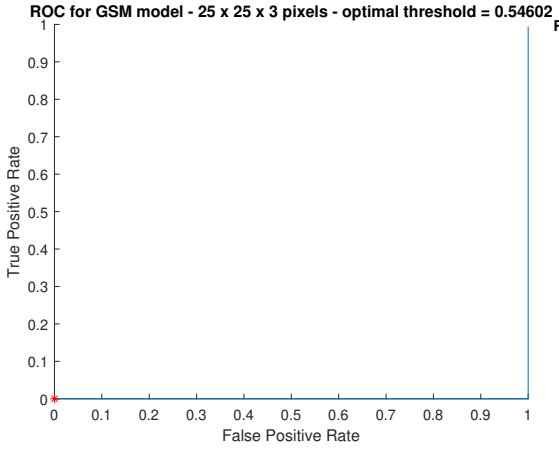

Figure 3: ROC for the $n = 25$ image Gaussian single model - bad model

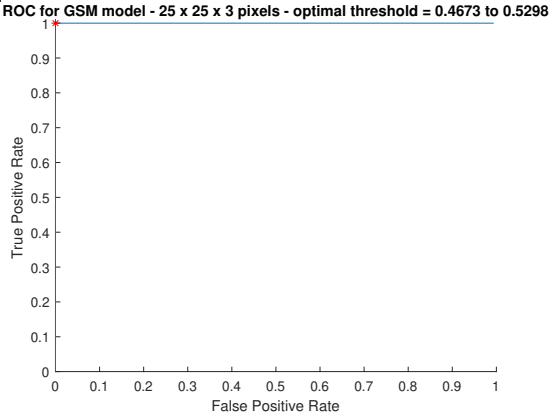

Figure 4: ROC for the $n = 25$ image Gaussian single model - good model

It was interesting to see the results of the model with resolution of 25. There is a high chance that I am calculating the ROC curve incorrectly, but my other ROC curves look reasonable so I am unsure. In the case that I am not, the model shown in Figure 4 perfectly can perfectly distinguish faces from non-faces at a resolution of 25. But the model seemed to be a hit or miss, since sometimes when seeing the results it would always guess wrong, like seen in the model in Figure 3. Either it had a large range for the threshold to maximize the true positives and minimize false positives, or the model spat out values with an always greater number of false positives.

I would have tried to show larger images, but anything larger take a long time to run (as a $60 \times 60 \times 3$ image will have dimensionality of 10800. The covariance matrix used will have to have 116,640,000 elements, and on a laptop PC one will encounter memory problems).

3 Mixture of Gaussian Distributions

3.1 Algorithm

The Gaussian mixture model is uses a multiple multivariate normal distributions to best estimate where our data lies within some higher dimensional subspace. The multivariate Gaussian pdf is defined similarly to (2)

$$p(\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} \left|\boldsymbol{\Sigma}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\right) \tag{5}$$

where $\mathbf{X}$ is a vector of N 1st dimensional data points $\mathbf{x}_n$ ($\mathbf{X} \in \mathbb{R}^{N \times D}, \mathbf{x}_n \in \mathbb{R}^D$), $\boldsymbol{\mu}$ is the mean vector of the data points ($\boldsymbol{\mu} \in \mathbb{R}^D$), and $\boldsymbol{\Sigma}$ is the covariance (positive semi-definite) matrix of the data set($\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$). For the GMM, multiple of these multivariate distributions are created and fit to the data set. This number will be denoted by k, for a total of K clusters. We can reconfigure (5) to add this higher dimension K toward the Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. To find the MLE for each multivariate cluster, we can start by using the posterior on observation $\mathbf{X}$

$$p(\textbf{face} \mid \mathbf{X}, \theta) = \frac{p(\mathbf{X}, \textbf{face} \mid \theta)}{p(\mathbf{X})} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} = r_{nk} \tag{6}$$

where θ contains the GMM parameters $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \pi\}$, **face** is a binary indicator on whether or not an image is classified as a face, and π is a vector of normalized weights. Conceptually, each π_k determines how much the k multivariate cluster contributes to the overall model. This posterior probability of any observed $\mathbf{x}_n$ is denoted by the scalar r_{nk}. Meaning, we evaluate the data point $\mathbf{x}_n$ on each cluster to find the magnitude r_{nk}. This value is used in updating our means and variances of each randomly initialized cluster to "hone" in on a likely distribution. This process is known as the expectation-maximization (EM) algorithm. The process indicated above has already concluded the E-step.

The M-step, or the maximization of the expectation by finding optimal parameters θ, is derived by finding the MLE w.r.t. each parameter. For the multivariate normal distribution, these values are given by

$$\pi_k^{[t+1]} = \frac{\sum_{n=1}^{N} r_{nk}}{\sum_{l=1}^{K} \sum_{n=1}^{N} r_{nl}} \tag{7}$$

$$\boldsymbol{\mu}_k^{[t+1]} = \frac{\sum_{n=1}^{N} r_{nk} \mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}} \tag{8}$$

$$\boldsymbol{\Sigma}_k^{[t+1]} = \frac{\sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_n - \boldsymbol{\mu}_k^{[t+1]})^\top}{\sum_{n=1}^{N} r_{nk}} \tag{9}$$

By alternating the E and M steps, we update our posterior probabilities as well as our weights, means, and variances to get a better fit of our GMM to our data. The E and M steps are then alternated until there is little-to-no change in our parameters θ.

3.2 Results

The covariance matricies $\boldsymbol{\Sigma}$ continued to collapse or erupt. And in my efforts to keep $\boldsymbol{\Sigma}$ a positive-definite matrix (for the calculations to be proper), I kept losing accuracy of my model. Just 3 EM steps would make the model stall at some, if not all, $\boldsymbol{\mu}$ values and my covariance matricies would be useless. Therefore, everything seen below within this section is only after 1 or 2 EM steps.

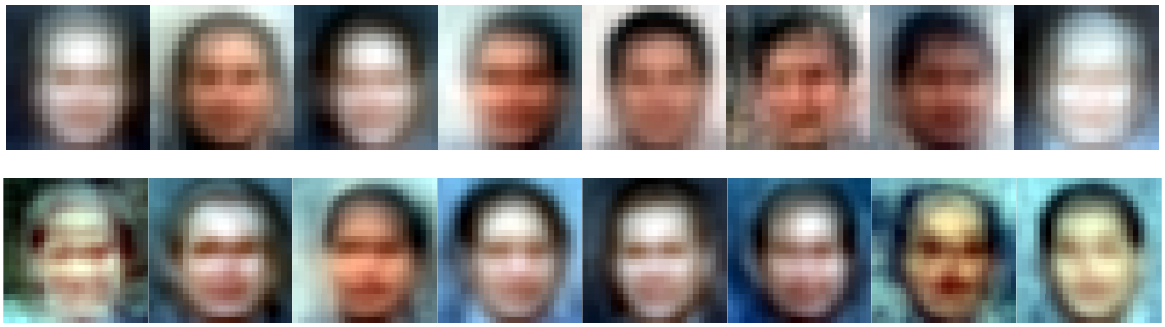

Figure 5: Means for $n = \{20, 25\}$ respectively. Images trained on a 8 cluster GMM

0.1268 0.1695 0.1763 0.0663 0.0563 0.1457 0.1664 0.0928

Figure 6: Means for $n = 25$. Images trained on a 8 cluster GMM. Weights shown underneath

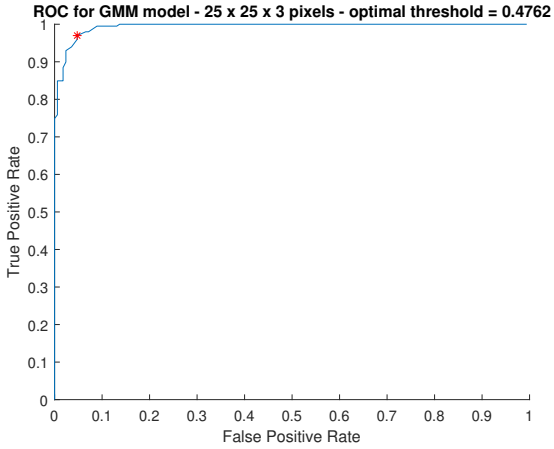

Figure 7: Using uniform weights - ROC for the $n = 25$ image Gaussian mixture model corresponding with Fig. 6

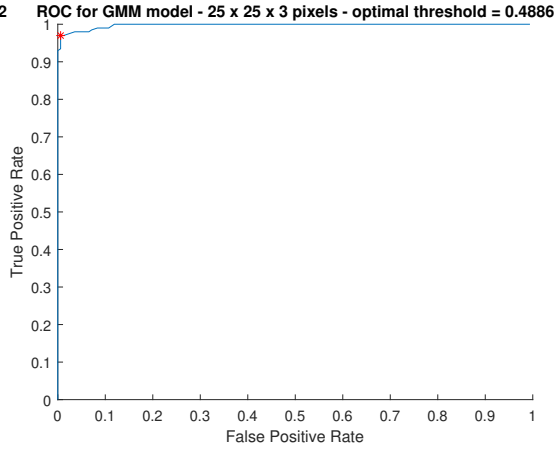

Figure 8: Using π_k weights - ROC for the $n = 25$ image Gaussian mixture model corresponding with Fig. 6

I accidentally computed the ROC for this model without incorporating the weights, and when I fixed it to include the weights I noticed something interesting — that when including the weights the model performs better! You can see that the optimal threshold to maximize true positives and minimize false positives increases from the time there were uniform weights applied versus when there were

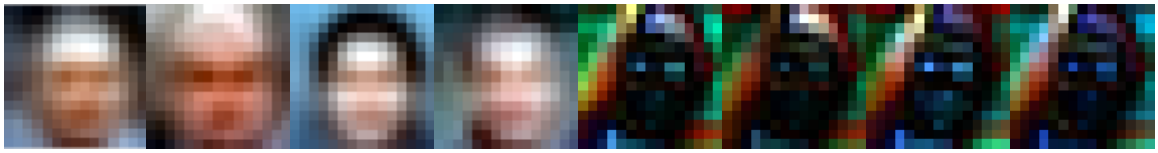

Figure 9: Means and diagonal covariances respectively for $n = 15$. Images trained on a 4 cluster GMM. Just showing using to display covariance matricies.

0.1189 0.1584 0.3298 0.1842 0.0453 0.0310 0.1012 0.0313

Figure 10: Means and diagonal covariances respectively for $n = 10$. Images trained on a 8 cluster GMM. Weights shown underneath

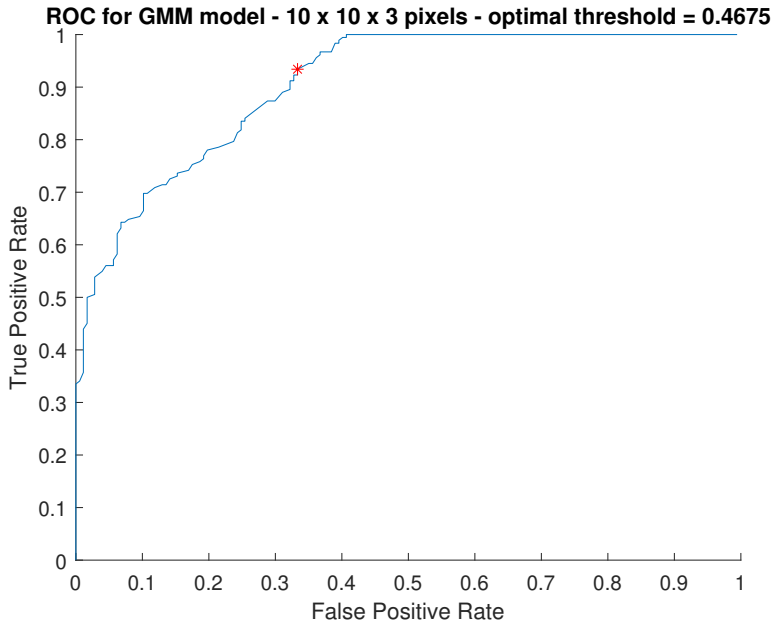

Figure 11: Using π_k weights - ROC for the $n = 10$ image Gaussian mixture model

4 Student's t-Distribution as marginalization

4.1 Algorithm

A single t-distribution can be used to model our image data by finding the MLE w.r.t. parameters μ, Σ, as well as ν degrees of freedom. Unfortunately there is no closed form for the MLE of the t-Distribution, but it can be calculated using an EM approach. To calculate the expectation of the distribution over each hidden variable h for the E-step, we result in

$$\mathrm{E}[h_n] = \frac{\nu + D}{\nu + (\mathbf{x}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu})} \tag{10}$$

where D is the number of dimensions. We then use this result in our M-step to maximize the likelihood function across parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

$$\boldsymbol{\mu}^{[t+1]} = \frac{\sum_{n=1}^{N} \mathrm{E}[h_n]\mathbf{x}_n}{\sum_{n=1}^{N} \mathrm{E}[h_n]} \tag{11}$$

$$\boldsymbol{\Sigma}^{[t+1]} = \frac{\sum_{n=1}^{N} \mathrm{E}[h_n](\mathbf{x}_n - \boldsymbol{\mu}^{[t+1]})(\mathbf{x}_n - \boldsymbol{\mu}^{[t+1]})^\top}{\sum_{n=1}^{N} \mathrm{E}[h_n]} \tag{12}$$

To optimize the degrees of freedom ν, a linear sweep can be done on the variable. However, for this project we simply chose the value of ν, which is sub-optimal but still works.

4.2 Results

Figure 12: Means and diagonal covariances for $n = 25$, trained on face and non-face images for a single t-Distribution model

The single t-Distribution model using the gamma function $\Gamma[\cdot]$ for evaluating the effectiveness of the model. I run into a computational error for this higher dimensional model, where I calculate

$$\Gamma\left[\frac{(\nu + D)}{2}\right]$$

where $D =$ the number of dimensions, which in the case above is $25 \times 25 \times 3 = 1875$. This results in an infinite solution in MATLAB, so my ROC curve fails to evaluate the model properly. I chose ν to be a low value of 2.1 (lowest possible being 2), and still the calculation fails. If I lower the image resolution to only $10 \times 10 \times 3$ and ν remains 2.1, the equation above evaluates to 7.3413×10^{262}, which is quite large. I do not know how to fix this.

5 Mixture of Student's t-Distribution

5.1 Algorithm

Similar to the previous section, the mixture of Student's t-distribution acts like a GMM, but with a t-Distribution instead. The hyperparameter ν can be tuned to maximize our already maximized likelihood function, but for the sake of computation time ν is a randomly selected constant greater than 2. A π_k term is introduced, similar to equation (6), but instead of Normal distributions we use the multivariate Student's t-distribution

$$Pr(\mathbf{x}) = f(\mathbf{x};\, \boldsymbol{\mu}_k,\, \boldsymbol{\Sigma}_k, \nu_k) = \frac{\Gamma\left[\frac{\nu+D}{2}\right]}{(\nu\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}\Gamma\left[\frac{\nu}{2}\right]} \tag{13}$$

where we can define

$$f(\mathbf{x}, \boldsymbol{\Psi}) = \sum_{l=1}^{K} \pi_l f(\mathbf{x};\, \boldsymbol{\mu}_l,\, \boldsymbol{\Sigma}_l, \nu_l) \tag{14}$$

to make a new term γ to be used in each E-step

$$\gamma_{nk} = \frac{p(\mathbf{X}, \textbf{face} \mid \mathbf{\Psi})}{p(\mathbf{X})} = \frac{\pi_k f(\mathbf{x}_n; \boldsymbol{\mu}_k, \mathbf{\Sigma}_k, \nu_k)}{f(\mathbf{x}_n, \mathbf{\Psi})} \tag{15}$$

the M-steps change slightly from equations (11) and (12), where the new steps include γ. In addition, the weights π_k are also updated

$$\pi_k^{[t+1]} = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N} \tag{16}$$

$$\boldsymbol{\mu}_k^{[t+1]} = \frac{\sum_{n=1}^{N} \gamma_{nk} \mathrm{E}[h_n] \mathbf{x}_n}{\sum_{n=1}^{N} \gamma_{nk} \mathrm{E}[h_n]} \tag{17}$$

$$\mathbf{\Sigma}_k^{[t+1]} = \frac{\sum_{n=1}^{N} \gamma_{nk} \mathrm{E}[h_n] (\mathbf{x}_n - \boldsymbol{\mu}^{[t+1]})(\mathbf{x}_n - \boldsymbol{\mu}^{[t+1]})^\top}{\sum_{n=1}^{N} \gamma_{nk}} \tag{18}$$

5.2 Results

Figure 13: Means for $n = 10$. Images trained on a 4 cluster TMM

Similar to the GMM, going through multiple EM steps caused my TMM means to converge. The covariance diagonal matrix calculation updates after each M-step aren't the most effective, however for only 1 step we get a decent range of the means. In addition, see the error given in the single t-Distribution model which prevents me from evaluating the model to get proper ROC curves.

6 Factor Analysis as marginalization

6.1 Algorithm

Factor analysis essentially sweeps the variance $\mathbf{\Sigma}$ across a higher dimensional plane $\mathbf{\Phi}$. This means that by using just the mean $\boldsymbol{\mu}$ and adding/subtracting $\mathbf{\Phi}$, the model moves across the main higher dimensional axis of the variance, so that multiple faces can be modeled by simple adding/subtracting values from the mean. A Normal distribution is used in evaluating this quantity, since

$$Pr(\mathbf{x}) = \mathcal{N}_{\mathbf{x}}(\boldsymbol{\mu}, \mathbf{\Phi}\mathbf{\Phi}^\top + \mathbf{\Sigma}) \tag{19}$$

Note to professor and/or TA's: I am already turning this in 2 days late, so I did not complete this section! I am sorry

7 Heatmaps

For fun, I decided to sweep a window across some images with different models to see how well they evaluate faces. After seeing the results, I can confirm that there is high probability I am calculating the ROC curves incorrectly because these results do not look promising. Nevertheless, these models are extremely simple so I was not expecting the results to be at all promising.

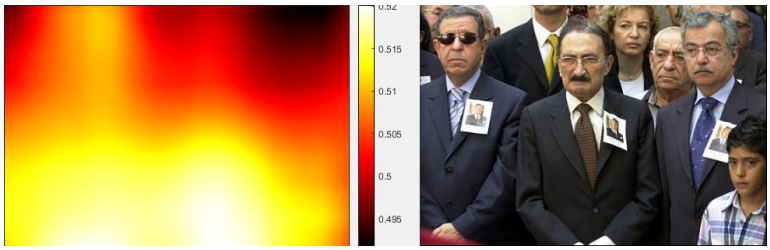

Figure 14: Sweeping window to test GSM model, trained on $n = 25$

As you can see, the results are not good at all. Here are some more

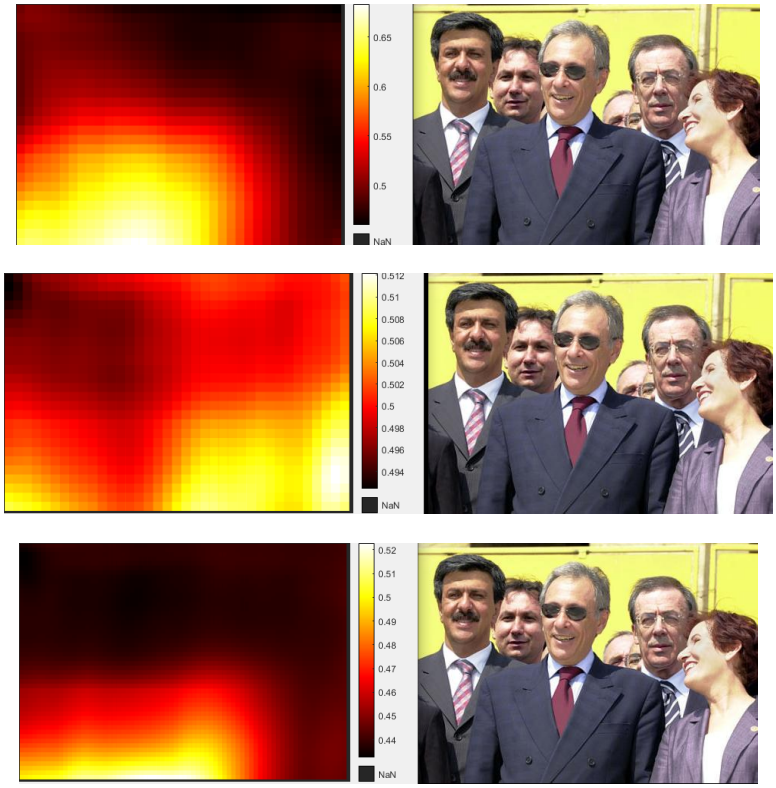

Figure 15: Sweeping window to test GSM models, and one TSM model respectively ($n \in \{25, 10, 20\}$)

8 Conclusion

Most of these models are very computationally intensive, so a regular PC is not effective in evaluating images. The Gaussian models are the easiest to implement, the t-Distribution models increase in accuracy and are less susceptible to outliers, and the factor analysis model helps increase computational efficiency due to a decrease in dimension dependability.